Available online at www.sciencedirect.com
ScienceDirect
Procedia Social and Behavioral Sciences 20 (2011) 806–815
Procedia Social and Behavioral Sciences

14th EWGT & 26th MEC & 1st RH

Scheduling algorithms for the lock scheduling problem

Jannes Verstichel[a,b,*], Patrick De Causmaecker[b], Greet Vanden Berghe[a]

[a]CODeS, KAHO Sint-Lieven,Gebroeders De Smetstraat 1, 9000 Gent, Belgium
[b]CODeS, KU Leuven Campus Kortrijk, Etienne Sabbelaan 53, 8500 Kortrijk, Belgium

Abstract

In the present contribution we will show that scheduling a lock having at least two identical chambers requires solving the identical parallel machine scheduling problem with unit processing times, release dates and sequence dependent setup times. The lock scheduling problem considers the order in which a number of ships should be transferred through a lock. A lock may have one or more parallel chambers of a different size, each requiring a certain amount of time to transfer any possible feasible set of ships, called a lockage, through the lock. A mathematical model and meta heuristic are proposed for solving this problem.

Selection and/or peer-review under responsibility of the Organizing Committee.
Keywords: Lock scheduling, parallel machine scheduling, (meta) heuristics

1. Introduction

Sea ships often have to pass one or more locks in order to enter a port. These locks guarantee a constant water level at the docks simplifying both loading and unloading of ships. Similarly, inland ships often pass several locks when travelling on waterways. These inland locks keep the water at both sides of the lock at a constant, navigable level.

A lock has at least one chamber, but may consist of multiple parallel chambers of different dimensions. Each chamber has a limited capacity and lockage duration, i.e. the time needed to change the water level in the chamber from one side to the other. Chambers should preferably not go in lockage empty or with only a small part of their capacity used, as this could result in excessive water usage and a possible disruption of the water levels. Changing the levels too infrequently, however, will result in very long waiting times for many ships, resulting in high costs for the shipping companies. The importance of limiting the water usage of a lock may depend on external factors such as the season e.g. increased flow due to heavy rainfall, etc.

Lock scheduling characteristics depend on the type of ship to be transferred. Whereas for sea ships the lockage time depends on both size and maneuverability, the lockage time for inland ships on the other hand, can be

* Corresponding author. Tel.: +32 (0)9 265 87 04; fax: +32 (0)9 225 62 69.
E-mail address: jannes.verstichel@kahosl.be .

1877–0428 © 2011 Published by Elsevier Ltd. Open access under CC BY-NC-ND license.
Selection and/or peer-review under responsibility of the Organizing Committee
doi:10.1016/j.sbspro.2011.08.089

considered constant. Both cases, however, share the very same ship placement problem, even though different approaches for scheduling the lockages are required.

The lock scheduling problem, besides scheduling, also considers some bin packing objectives and constraints. It therefore belongs to the category of structured problems, encompassing characteristics of more than one (known) combinatorial optimization problem.

There are only few publications concerning the lock scheduling problem in academic literature. The optimal sequencing of tows/barges for single chamber locks with set-up times (Nauss, 2008) allows one tow/barge to be transferred at a time. It considers all tows/barges present at the lock before the first lockage, while reducing the water usage is not an issue. The applicability of different queuing models for lock capacity analysis (Wilson, 1978) shows that good queuing models exist for single chamber locks, but not for locks with parallel chambers. Different congestion solving strategies for the Upper Mississippi river are discussed by Campbell et al. (2007) and Campbell et al. (2009). On that river, barges are joined together into tows for transport, which need to be transferred by single chamber locks that are often smaller than the tow itself. The tow is split into different groups of barges and these groups are transferred one at a time, and are then rejoined for the next phase of their travel. Different strategies are considered for increasing the throughput of the locks and a simulation tool is built for validating the strategies. The lock scheduling problem with multiple parallel chambers was introduced by Verstichel and Vanden Berghe (2009). A problem specific heuristic is used to place ships in a chamber, and lockages are sequenced in a first come first served way. These initial results are improved by applying a late acceptance multiple neighborhood meta heuristic.

In the following contribution, different types of lock scheduling problems with multiple parallel chambers are considered from the scheduling point of view. In the first place, such an approach entails the presentation of a mathematical model and a performance analysis for the scheduling problem. Later on, a meta heuristic approach for solving the problem is presented. In both stages, a high performance packing heuristic for placing ships in chambers (Verstichel et al. 2011) is used.

2. Problem identification

As indicated in the introduction, the lock scheduling problem can be decomposed into a bin packing sub problem and a scheduling sub problem. The scheduling sub problem can be solved by assigning lockages to chambers. The chambers are the physical components of the lock in which ships are transferred from one level of the waterway to the other. Each chamber is defined by its type and initial state. The chamber's type determines its dimensions (length and width), lockage duration (i.e. the time the chamber needs to change its water level) and the set of ships that can be transferred by the chamber. The chamber's initial state defines the first available time of the chamber, and its state at that time (upstream/downstream). A lock may have multiple chambers, several of which can be of the same type. Each chamber type has two associated ship lists (one for each direction) which contain the ships that have to be transferred by the chambers of this type. We call a feasible configuration of ships with respect to the size of the chamber type a lockage. A lockage also has a direction and an earliest processing time, both depending on the ships in the lockage. The initial assignment of ships to the ship lists is based on the width of the ships, but can be changed during the solution optimization. An example of a lock scheduling problem is depicted in Figure 1. This problem has two identical chambers, six ships travelling upstream and seven ships travelling downstream.

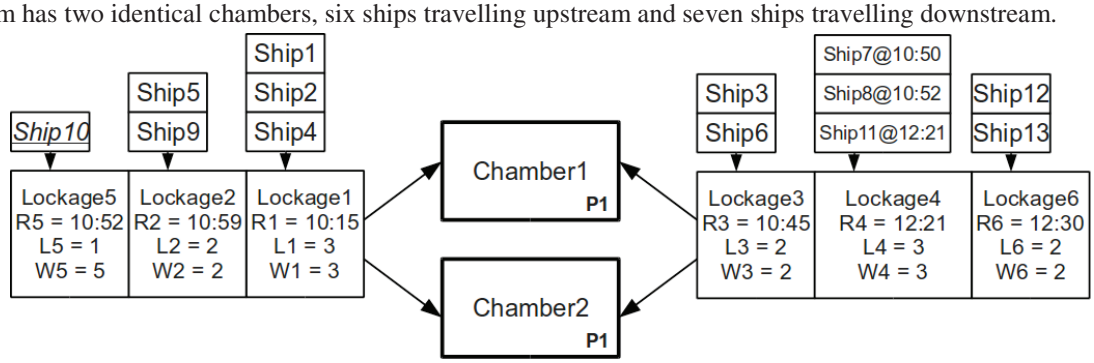

Figure 1. Visual example of a lock scheduling problem with two identical parallel chambers.

From a scheduling point of view, a chamber can be considered as a machine, with chambers of the same type corresponding to identical machines. A lockage can be seen as a job, where each job can be processed on any machine of a particular type. As each lockage can be transferred by one chamber type only, the scheduling sub problem can be identified as an identical parallel machine scheduling problem. When the lock has chambers of different types, this problem will have to be solved for each chamber type individually.

With respect to the processing times, two different cases need to be considered depending on the type of ship that needs to be transferred. When only inland ships need to be transferred, all processing times will be equal, since we can consider the time needed for ships to enter and exit the chamber to be constant. The situation changes when sea ships are taken into consideration. As these ships may require a long time to enter and exit a chamber depending on their size and on the chamber's dimensions, the processing times will depend on the ships present in the lockage.

A lockage cannot be processed before all the ships that have been assigned to that lockage have arrived at the lock. Therefore, release dates have to be added to the model.

Due to physical limitations, the only way to change a chamber's state is by processing a lockage. Therefore, consecutive lockages in the same chamber should be in opposite directions, and the addition of sequence dependent setup times to our model is required: There is no setup time between lockages in opposite directions and a setup time equal to the lockage time of the chamber type between any lockages in the same direction when they are transferred by the same chamber.

Using this information, the scheduling part of the lock scheduling problem for inland ships can be identified as the identical parallel machine scheduling problem with unit processing times, release dates and sequence dependent setup times or $P|r_i, p_i = p, s_{ij}|\sum w_i T_i$. When sea ships have to be transferred by the lock, the unit processing times no longer apply.

In order to schedule lockages, ships must be assigned to them first. The set of ships that is assigned to a lockage influences the schedule because the ship arrival times determine the release date of the lockage. Using the example in Figure 1, a strong reduction of the waiting time for ships 7 and 8 (lockage 4) could be obtained by forcing ship 11 into a later lockage (for example together with ships 12 and 13). It is clear that the scheduling algorithm should be able to influence the ship assignments to lockages.

The packing heuristic used (Verstichel et al., 2011) employs a first come first served policy based on the position of the ships in the ship lists. This way, the ship to lockage assignments can be altered by simply changing the ship lists of the different chamber types.

Two different approaches to scheduling-packing interaction are presented, depending on the properties of the scheduling algorithm.

3. Mathematical model

The scheduling sub problem of the lock scheduling problem was identified as an identical parallel machine scheduling problem with unit processing times, release dates and sequence dependent setup times. As mentioned, solving the sub problem requires solving this parallel machine scheduling problem for each chamber type of the lock. A mathematical model based on the mixed integer linear programming model of Balakrishnan et al. (1999) is explained in detail below. In a lock scheduling setting, their model requires an additional objective term, namely maximum tardiness. By combining this additional objective term with a precedence constraint (see constraint 9), we are able to obtain a 'fair' optimal solution, i.e. the solution with the smallest deviation from the first come first served policy with respect to the arrival times of the ships. These additions are necessary because shippers do not tolerate excessive waiting times for one individual.

Variables:
x_{ij} : lockage i precedes lockage j on the same machine
y_{ik} : lockage i is transferred by chamber k
t_i : tardiness of lockage i
c_i : completion time of lockage i
T : maximum tardiness

Constants:
N : the number of lockages
M : the number of chambers (of the current chamber type)
r_i : release date of lockage i
d_i : due date of lockage i
w_i : weight of lockage i, depends on the number of ships that are assigned to lockage i, and their priority
l_i : size of lockage i, i.e. the number of ships that are assigned to lockage i
s_{ij} : setup time between lockage i and lockage j
p : processing time of a lockage
K_T : relative importance of maximum tardiness
L : large positive number

$$\min \sum_{i=1}^{N} w_i t_i + \frac{T}{K_T} \qquad (1)$$

s.t.

$$\sum_{k=1}^{M} y_{ik} = 1, \forall i = 1,...,N \qquad (2)$$

$$y_{ik} + \sum_{\substack{l=1 \\ l \neq k}}^{M} y_{jl} + x_{ij} \leq 2, \forall i = 1,...,N-1; \forall j = i+1,...,N; k = 1,...,M \qquad (3)$$

$$c_j - c_i + L\left(3 - x_{ij} - y_{ik} - y_{jk}\right) \geq p + s_{ij}, \quad \forall i = 1,...,N-1; \forall j = i+1,...,N; k = 1,...,M \qquad (4)$$

$$c_i - c_j + L\left(2 + x_{ij} - y_{ik} - y_{jk}\right) \geq p + s_{ji}, \quad \forall i = 1,...,N-1; \forall j = i+1,...,N; k = 1,...,M \qquad (5)$$

$$c_i - t_i = d_i, \forall i = 1,...,N \qquad (6)$$

$$c_i \geq r_i + p, \forall i = 1,...,N \qquad (7)$$

$$T \geq t_i, \forall i = 1,...,N \qquad (8)$$

$$c_j \geq c_i, \forall i, j : r_i < r_j, l_i = l_j, s_{ij} = p \qquad (9)$$

$$t_i \geq 0, \forall i = 1,...,N \qquad (10)$$

$$c_i \geq 0, \forall i = 1,...,N \qquad (11)$$

$$x_{ij} \in \{0,1\}, \forall i = 1,...,N-1; \forall j = i+1,...,N \qquad (12)$$

$$y_{jk} \in \{0,1\}, \forall j = 1,...,N; \forall k = 1,...,M \qquad (13)$$

The objective (1) minimizes the total weighted tardiness and the maximal tardiness. The weight of each lockage depends on the number of ships that are transferred by this lockage and their priority. Using the example from Figure 1 we can see that lockage 1 contains only normal ships, resulting in a weight that is equal to its size. Lockage 5 on the other hand, contains a priority ship that is 5 times as important as a normal ship, making the lockage's weight equal to 5.

Constraint (2) expresses that each lockage is processed by exactly one chamber, while constraint (3) makes sure that variable x_{ij} can only take the value one when lockages i and j are assigned to the same chamber. This way a

strong reduction of the number of binary variables is possible, as M times less binary x_{ij} variables are needed, compared to using x_{ijk}.

Constraints (4) and (5) are disjunctive and establish the relationship between the completion times of two lockages i and j iff both lockages are assigned to the same chamber.

For a more detailed explanation of constraints (3), (4) and (5) we refer to the paper of Balakrishnan et al. (1999).

The tardiness of each lockage is calculated using constraint (6). This constraint is valid, as the due date of a lockage is equal to its release date r_i incremented with the processing time p of the chamber type. Constraint (7) guarantees that the release dates of the lockages are taken into account, while constraint (8) calculates the maximum tardiness over all lockages. Constraint (9) adds the first come first served rule with respect to the release dates of the lockages between all lockages that travel in the same direction and transfer the same number of ships. This constraint adds fairness to the model, making sure each lockage will be scheduled as close to its release date as possible, even if scheduling it later would not affect the objective. When looking at the example from Figure 1, this constraint will ensure that lockage 3 is processed before lockage 6.

Finally, constraints (10) and (11) denote that the real variables are non-negative and (12) and (13) define the binary variables.

While this model is able to solve the scheduling problem to optimality, it cannot directly influence the ship to lockage assignments. By changing the ship lists based on the waiting times of each individual ship and the total waiting time induced by each lockage, better ship to lockage assignments, with respect to the waiting times, could be obtained by the packing algorithm. For example, ships could be assigned to a later lockage, despite sufficient spare room in the previous lockage towards the water level aimed at. This could be the case when the arrival time of the ship is much later than the arrival time of the other ships in the lockage. Assigning ships or entire lockages to another chamber type in order to reduce congestion for a certain chamber type is another way to influence the solution quality. However, due to the long calculation times required for solving this mixed integer linear programming model (Section 5) this approach could not be investigated any further.

4. Meta heuristic approach

Next to the optimal MILP approach, we also developed a ‘simple random’ meta heuristic. This heuristic randomly selects one neighborhood at each iteration of the search and performs one move from this neighborhood, hence the name ‘simple random’. The following neighborhoods are used to influence the ship lists of the different chamber types in order to change the solution that is obtained by the underlying packing heuristic:

- SwapShips: two ships in the ship lists are swapped, $O(N^2)$
- ShiftShip: change the position of a single ship within one ship list, $O(N^2)$
- ChangeChamberType: move a ship to another ship list, changing its chamber type assignment, $O(MN^2)$
- AddDummyShip: add a DummyShip to a shiplist, forcing the packing heuristic to close the current lockage
- RemoveDummyShip: remove a DummyShip from a shiplist

Evaluating any of the first three neighborhoods entirely is very time consuming, especially when the problem size increases. Different approaches exist to limit the number of moves that are evaluated at each iteration, without losing the possibility to obtain good results. Using tournament selection, a limited number of moves will be generated randomly for the selected neighborhood and the best of these moves will be applied to the solution. This approach is very fast, but cannot exploit any knowledge of the problem characteristics. Another possibility is the usage of the corridor method (Sniedovich and Voss, 2006). A time based corridor will reduce the size of the neighborhood by considering only the swaps and moves when the difference in arrival time between the affected ship(s) and their new neighbors does not exceed a certain threshold, thus only generating interesting moves. Although this method reduces the size of the neighborhoods in a very intelligent way, they may still be quite large when taking into account that a new packing must be calculated for every move. A combination of the corridor method and tournament selection tackles the disadvantages of both approaches, while keeping all of their benefits. By performing a tournament selection on the neighborhood after the corridor method has been applied, a small number of interesting moves is generated, allowing a fast and intelligent search. The difference between the three approaches is visualized in Figure 2.

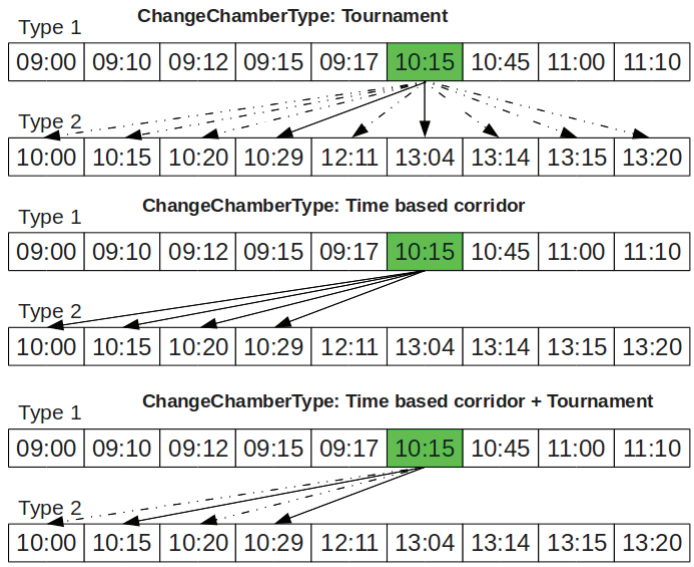

Figure 2. Behaviour of the different approaches to reducing the size of the neighborhoods. Dashed lines represent the possible moves, full lines the selected ones.

5. Experimental results

The proposed scheduling approaches were tested on both randomly generated test instances and a test instance generated from historical data. The generated test instances from Verstichel and Vanden Berghe (2009) are available online. The instance based on historical data contains 2 different problems. The properties of the instances are added in Table 1. The exact solutions were obtained using Gurobi 4.0.1 under academic license, the meta heuristic was implemented using Java SDK 1.6.

Table 1. Properties of the test instances.

Traffic properties	Generated instances	Real-life instances
Mean inter arrival time (min)	5, 10, 15, 30	20, 10
Number of ships	20, 100, 1000	258, 821
Upstream/Downstream traffic	50/50, 30/70	53/47, 52/48
Lockage duration (min)	16	16

Lock properties	Width (m)	Length (m)	#Chambers
ChamberType 1	16	136	2
ChamberType 2	24	200	1

5.1. Initial solution

An initial solution is constructed for each test instance based on the width of the ships. Each ship is assigned to a single chamber type based on the ratio of its width to that of the chamber. Each ship is assigned to the smallest chamber for which this ratio is smaller than the given WidthRatio.

5.2. Exact approach

The results on the smallest test instances (20 ships) show that the MILP model is able to decrease the total waiting time of the ships with up to 41%, with an average of 7%, compared to using a simple first come first served

scheduling approach. While this model requires less than 10 seconds to solve a single scheduling iteration for the smallest test instances, the solver timed out at 15 hours for most of the medium sized ones (100 ships). Due to these unacceptably long calculation times no further experiments were performed with the exact solution approach.

5.3. Meta heuristic approach

With respect to the simple random meta heuristic, the combination of tournament selection and the time based corridor was used to intelligently decrease the size of the neighborhoods. The parameters used for the meta-heuristic search are added in Table 2. The RangeFactor parameter is multiplied by the average inter arrival time of the problem to obtain the corridor range. A summary of the test results for a width fraction of 0.5 are shown in Table 3. From this table we can see that there is a significant difference in solution quality between the initial solution (first-come-first-served) and the solution after optimization, both with respect to the number of lockages and the waiting times. Similar results are obtained for all instances and all parameter settings. No significant differences can be found between the different tournament factors (Table 5), while the heuristic does seem to favor the larger RangeFactor values (Table 3). Using a RangeFactor of either 5 or 11 is significantly better than using a RangeFactor of 3 (p-value = 0.043 and 0.023 respectively). However, within one RangeFactor setting, none of the tournament values proved to be significantly better than the others, and each tournament value obtained at least one best result.

With respect to the calculation times, we can see from Table 4 that a larger tournament factor results in lower calculation times. This difference is significantly larger for the small instances compared to the large ones (42% and 25% difference respectively). The calculation times do however stay below 3 minutes for all instances. The complete results showed that the RangeFactor variable doesn't have a noticeable influence on the calculation time.

Comparable results are obtained when applying the meta heuristic to the real life instances (Table 6).

Table 2. Parameters for the meta heuristic search.

Parameters	
Solution evaluations	8192
RangeFactor	3, 5, 11
Tournament factor	1, 2, 4, 8, 16, 32, 64
WidthFraction	0.0, 0.5, 1.0
Lockage cost	10000
Waiting cost normal	1
Waiting cost priority	10

Table 3. Solution cost for the different RangeFactors.

		Range 3		Range 5		Range 11	
Instance	**FCFS**	**Worst**	**Best**	**Worst**	**Best**	**Worst**	**Best**
R-5-20-0.3	90674	90397	90397	80319	**80221**	80319	**80221**
R-5-20-0.5	100456	80343	60468	80291	60462	60489	**50392**
R-10-20-0.3	90735	90167	70493	90231	70493	80324	**70344**
R-10-20-0.5	70770	70417	70242	70417	60743	70348	**60473**
R-15-20-0.3	90790	90329	80323	80352	70739	80447	**70477**
R-15-20-0.5	81255	81102	80983	80949	80863	80949	**80808**
R-30-20-0.5	92557	81121	61324	80789	61268	70947	**61035**
R-30-20-0.3	131002	120451	100777	120451	100777	120451	**100773**
R-5-100-0.5	264302	232563	201997	242360	**192476**	242323	192591
R-5-100-0.3	334255	312172	282437	301913	272382	292732	**272118**
R-10-100-0.5	306679	263541	204271	243963	**194486**	244131	223677
R-10-100-0.3	327019	323344	263293	283658	**262913**	293235	263420
R-15-100-0.3	370293	334571	294195	334103	304205	314747	**264863**
R-15-100-0.5	318765	296218	275780	287334	**246970**	284920	254591
R-30-100-0.5	290694	282225	245104	264498	245911	281154	**235245**
R-30-100-0.3	341291	301491	283917	309981	**274253**	312167	279065
R-5-1000-0.5	2539708	2438786	**2357166**	2470234	2376183	2469078	2388469
R-5-1000-0.3	3093681	3000189	2877804	2979889	2839175	3008119	**2790620**
R-10-1000-0.3	3088698	2953624	2895618	2964393	**2819403**	2962412	2851919
R-10-1000-0.5	2640480	2461824	**2337813**	2454852	2376227	2439635	2356457
R-15-1000-0.5	2793050	2555545	2451980	2533703	2417599	2456512	**2381604**
R-15-1000-0.3	3123528	2990670	2895380	2955111	2868767	2966380	**2854645**
R-30-1000-0.5	2870770	2571033	2503169	2585353	**2473281**	2609229	2504350
R-30-1000-0.3	3307445	3149461	3050117	3176708	3053168	3177629	**3032352**

Table 4. Average calculation times in seconds for each instance size. (RangeFactor = 3)

	Tournament						
Instance size	64	32	16	8	4	2	1
20 ships	**1.07**	1.32	1.57	1.54	1.57	1.61	1.79
100 ships	**9.92**	10.14	10.53	11.51	11.69	12.36	12.51
1000 ships	**121.40**	127.87	139.59	143.60	146.07	151.79	162.00

Table 5. Results for the different TournamentFactors, best results in Bold (RangeFactor = 3)

	Tournament						
Instance	64	32	16	8	4	2	1
R-5-20-0.3	**90397**	**90397**	**90397**	**90397**	**90397**	**90397**	**90397**
R-5-20-0.5	**60468**	**60468**	**60468**	**60468**	80343	70327	60472
R-10-20-0.3	90167	70528	90167	**70493**	70528	**70493**	**70493**
R-10-20-0.5	70348	70385	70417	70385	70348	**70242**	70348
R-15-20-0.3	80333	80333	**80323**	**80323**	80339	80333	90329
R-15-20-0.5	**80983**	**80983**	81043	81043	81102	81043	81043
R-30-20-0.3	120451	100845	**100777**	100845	110554	100837	110554
R-30-20-0.5	71039	81121	71131	71363	**61324**	81032	71001
R-5-100-0.3	302034	292123	312172	301983	**282437**	292224	292627
R-5-100-0.5	222661	222516	222300	222258	232563	**201997**	222394
R-10-100-0.3	283662	**263293**	283528	263574	283449	323344	283811
R-10-100-0.5	225681	233334	**204271**	204851	263472	224504	263541
R-15-100-0.3	313876	**294195**	313462	314127	334571	313655	314029
R-15-100-0.5	287357	285446	286549	286734	285507	296218	**275780**
R-30-100-0.3	284034	**283917**	301491	290560	293126	294133	295227
R-30-100-0.5	282225	255554	245104	264582	263796	271984	**247038**
R-5-1000-0.3	3000189	2980394	2899995	2917470	2919269	2916801	**2877804**
R-5-1000-0.5	2393373	2428192	2438172	2357693	**2357166**	2408063	2438786
R-10-1000-0.3	2945261	2902087	2896486	2953624	**2895618**	2898664	2915441
R-10-1000-0.5	2443210	2418288	**2337813**	2452318	2436907	2395728	2461824
R-15-1000-0.3	2990670	2947101	2941368	2927949	**2895380**	2919649	2929408
R-15-1000-0.5	2555545	2456129	2507843	**2451980**	2509993	2489131	2475990
R-30-1000-0.3	3141672	3149461	3132612	3083649	3140002	**3050117**	3081737
R-30-1000-0.5	2571033	2533312	**2503169**	2509765	2535609	2506795	2527228

Table 6 : Results on the real life instances.

		Tournament						
Range	**Instance**	64	32	16	8	4	2	1
55 min	1	2565455	2504252	**2476326**	2538958	2562103	2558631	2603714
	2	749423	721485	760829	742998	752410	**714398**	759509
110 min	1	2511436	2585627	2524617	2520795	2506440	**2486312**	2526859
	2	**722046**	726621	743357	739649	760458	764094	725686
165 min	1	2510295	2546149	2503074	2508378	2506593	2524941	**2467405**
	2	**686993**	755792	765181	740793	721844	710124	767906

6. Conclusion and further research

We have presented two approaches for the scheduling sub problem of the lock scheduling problem. Besides, two different methods for the scheduling algorithms to influence the behavior of the packing algorithm were proposed. While it was impossible to solve medium and large problems in an acceptable time period using the MILP model,

results of the meta heuristic approach show that considerable improvements of the solution quality can be obtained in short computation times. A thorough analysis of experiments on both generated and real life instances did however show that there is no specific parameter setting that is significantly better than all others. Such a conclusion entails that the combination of a meta heuristic and a high performance packing heuristic is a promising technique for solving the lock scheduling problem, but further research is required in order to obtain the best results.

Further research will first focus on using other meta heuristics and determining the efficiency of each neighborhood. These results will be compared to the other results for the lock scheduling problem, available in literature. Furthermore, other approaches to the mathematical model will be investigated, in order to obtain better relaxations and decrease the computation time.

Acknowledgement

Research funded by a Ph.D. grant (SB091152) of the Agency for Innovation by Science and Technology (IWT).

References

Balakrishnan, N., Kanet, J.J. & Sridharan, V. (1999). Early/tardy scheduling with sequence dependent setups on uniform parallel machines. *Computers & Operations Research*, *26(2)*, 127-141.

Campbell, J.F., Smith, D.L., Seeweny II, D.C., Mundy, R. & Nauss, R.M. (2007). Decision Tools for Reducing Congestion at Locks on the Upper Mississippi River. *Proceedings of the 40th Hawaii International Conference on System Sciences*.

Campbell, J.F., Smith, D.L., Seeweny II, D.C. (2009). A Robust Strategy for Managing Congestion at Locks on the Upper Mississippi River. *Proceedings of the 42nd Hawaii International Conference on System Sciences*.

Nauss, R.M. (2008). Optimal sequencing in the presence of setup times for tow/barge traffic through a river lock. *European Journal of Operational Research*, *187*, 1268–1281.

Sniedovich, M., Voß, S. (2006). The corridor method: a dynamic programming inspired metaheuristic. *Control and Cybernetics, 35*, 551-578

Verstichel, J. & Vanden Berghe, G. (2009). A Late Acceptance Algorithm for the Lock Scheduling Problem. *Logistik Management*, 457–478

Verstichel, J., De Causmaecker, P. & Vanden Berghe, G. (2011). An adapted best fit heuristic for the lock scheduling problem. *Technical report*. Retrieved from http://allserv.kahosl.be/~jannes/index.html

Wilson, H.G. (1978). On the applicability of queueing theory to lock capacity analysis. *Transportation Research*, *12*, 175-180.